First-order Logic

Russell, Norvig, "Artificial Intelligence: A Modern Approach", 3rd ed

Outline

Why FOL?

Syntax and semantics of FOL

Using FOL

Wumpus world in FOL

Knowledge engineering in FOL

Pros and cons of propositional logic

☺ Propositional logic is declarative

☺ Propositional logic allows partial/disjunctive/negated information

- (unlike most data structures and databases)

☺ Propositional logic is compositional:

- meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$

☺ Meaning in propositional logic is context-independent

- (unlike natural language, where meaning depends on context)

☹ Propositional logic has very limited expressive power

(unlike natural language)

- E.g., cannot say "pits cause breezes in adjacent squares“
  - except by writing one sentence for each square

First-Order Logic

Whereas propositional logic assumes the world contains facts,

first-order logic (like natural language) assumes the world contains

- Objects: people, houses, numbers, colors, games, …
- Relations: red, round, prime, brother of, bigger than, part of, comes between, …
- Functions: father of, best friend, one more than, plus, …

Syntax of FOL: Basic elements

Constants	John, 2, DIT,...
Predicates	Brother, >,...
Functions	Sqrt, LeftLegOf,...
Variables	x, y, a, b,...
Connectives	$\neg$, $\Rightarrow$, $\wedge$, $\vee$, $\Leftrightarrow$
Equality	=
Quantifiers	$\forall$, $\exists$

Syntax of FOL

Sentence → AtomicSentence
| Sentence Connective Sentence
| Quantifier Variable Sentence
| ¬Sentence
| (Sentence)

AtomicSentence → Predicate(Term, Term, ...)
| Term=Term

Term → Function(Term,Term,...)
| Constant
| Variable

Syntax of FOL (cont.)

Connective → ∨
| ∧
| ⇒
| ⇔

Quanitfier → ∃ | ∀

Constant → A | John | Carl

Variable → x | y | z | ...

Predicate → Brother | Owns | ...

Function → father-of | plus | ...

Predicates vs. functions?

Constants represent individuals in the world

- Mary
- 3
- Green

Predicates map individuals to truth values

For example

- greater(5,3)
- green(Grass)
- color(Grass, Green)

Functions map individuals to individuals

- father-of(Mary) = John
- color-of(Sky) = Blue

Complex sentences

Complex sentences are made from atomic sentences using connectives

$$\neg S,\ S_1 \wedge S_2,\ S_1 \vee S_2,\ S_1 \Rightarrow S_2,\ S_1 \Leftrightarrow S_2$$

For example,

$Sibling(John,Richard) \Rightarrow Sibling(Richard,John)$

$>(1,2) \vee \leq (1,2)$

$>(1,2) \wedge \neg >(1,2)$

Truth in first-order logic

Sentences are true with respect to a model and an interpretation

Model contains objects (domain elements) and relations among them

Interpretation specifies referents for

constant symbols → objects

predicate symbols → relations

function symbols → functional relations

An atomic sentence $predicate(term_1,\ldots,term_n)$ is true
iff the objects referred to by $term_1,\ldots,term_n$
are in the relation referred to by *predicate*

Universal quantification $\forall$

$\forall$<variable> <sentence>

Example: Everyone at DIT is smart:

$\forall x\ At(x,DIT) \Rightarrow Smart(x)$

$\forall x\ P(x)$ is true in a model m iff P is true with x being each possible object in the model

Roughly speaking, this is equivalent to the conjunction of instantiations of P

$At(John,DIT) \Rightarrow Smart(John)$

$\wedge\ At(Richard,DIT) \Rightarrow Smart(Richard)$

$\wedge\ At(Mary,DIT) \Rightarrow Smart(Mary)$

$\wedge\ \ldots$

A common mistake to avoid

Typically $\Rightarrow$ is the main connective with $\forall$

Common mistake: using $\wedge$ as the main connective with $\forall$:

$\forall x\ At(x,DIT) \wedge Smart(x)$

means "Everyone is at DIT and everyone is smart"

Existential quantification ∃

∃<variables> <sentence>

Example: Someone at DIT is smart:

$\exists x\ At(x,DIT) \wedge Smart(x)$

$\exists x\ P(x)$ is true in a model m iff P is true with x being some possible object in the model

Roughly speaking, equivalent to the disjunction of instantiations of P

$At(John,DIT) \wedge Smart(John)$

$\vee\ At(Richard,DIT) \wedge Smart(Richard)$

$\vee\ At(Mary,DIT) \wedge Smart(Mary)$

$\vee\ \ldots$

Another common mistake to avoid

Typically $\wedge$ is the main connective with $\exists$

Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

$$\exists x\ At(x, DIT) \Rightarrow Smart(x)$$

is true if there is anyone who is not at DIT!

Well-formed formula (wff)

A well-formed formula (wff) is a sentence containing no "free" variables.

That is, all variables are "bound" by universal or existential quantifiers.

For example, $(\forall x)\ P(x,y)$

- has x bound as a universally quantified variable, but y is free
- not a wff

Properties of quantifiers

1. $\forall x\ \forall y$ is the same as $\forall y\ \forall x$
2. $\exists x\ \exists y$ is the same as $\exists y\ \exists x$
3. $\exists x\ \forall y$ is not the same as $\forall y\ \exists x$

$\exists x\ \forall y\ Loves(x,y)$
"There is a person who loves everyone in the world"

$\forall y\ \exists x\ Loves(x,y)$
"Everyone in the world is loved by at least one person"

Expressing ∀ and ∃ as each other

De Morgan's law:
The quantifiers ∀ and ∃ can be expressed as each other:

$$\forall x\ P(x) \equiv \neg\exists x\ \neg P(x)$$
$$\exists x\ P(x) \equiv \neg\forall x\ \neg P(x)$$

For example,

$\forall x$ Likes(x,IceCream) is logically equiv. to $\neg\exists x\ \neg$Likes(x,IceCream)
$\exists x$ Likes(x,Broccoli) is logically equiv. to $\neg\forall x\ \neg$Likes(x,Broccoli)

Equality

$term_1 = term_2$ is true under a given interpretation if and only if $term_1$ and $term_2$ refer to the same object

E.g., definition of *Sibling* in terms of *Parent*:

$$\forall x,y\ Sibling(x,y) \Leftrightarrow [\neg(x = y) \land \exists m,f \neg (m = f) \land Parent(m,x) \land Parent(f,x) \land Parent(m,y) \land Parent(f,y)]$$

Translating English to FOL

Every gardener likes the sun.
$(\forall x)$ gardener(x) $\Rightarrow$ likes(x,Sun)

You can fool some of the people all of the time.
$(\exists x)$ (person(x) ^ $(\forall$ t)(time(t) $\Rightarrow$ can-fool(x,t)))

You can fool all of the people some of the time.
$(\forall$ x) (person(x) $\Rightarrow$ $(\exists t)$ (time(t) ^ can-fool(x,t)))

All purple mushrooms are poisonous.
$(\forall$ x) (mushroom(x) ^ purple(x)) $\Rightarrow$ poisonous(x)

Translating English to FOL

No purple mushroom is poisonous.

$\neg(\exists x)$ purple(x) ^ mushroom(x) ^ poisonous(x)

or, equivalently,

$(\forall x)$ (mushroom(x) ^ purple(x)) $\Rightarrow \neg$poisonous(x)

There are exactly two purple mushrooms.

$(\exists x)(\exists y)$ mushroom(x) ^ purple(x) ^ mushroom(y) ^ purple(y) ^ ~(x=y) ^ $(\forall z)$ (mushroom(z) ^ purple(z)) $\Rightarrow$ ((x=z) v (y=z))

Mary is not tall.

$\neg$Tall(Mary)

Translating English to FOL: Kinship domain

Brothers are siblings

$\forall x,y\ Brother(x,y) \Rightarrow Sibling(x,y)$

One's mother is one's female parent

$\forall m,c\ Mother(c) = m \Leftrightarrow (Female(m) \wedge Parent(m,c))$

“Sibling” is symmetric

$\forall x,y\ Sibling(x,y) \Leftrightarrow Sibling(y,x)$

Inference in FOL

Inference rules for Predicate Logic apply to FOL as well

We can also use the following new sound inference rules for use with quantifiers:

- Universal elimination (also called Universal instantiation)
- Existential elimination (also called Existential instantiation)
- Universal generalization
- Existential generalization

Universal Elimination/ (Univ. Instantiation)

If $\forall x\ P(x)$ is true, then $P(A)$ is also true,
where A is a constant in the domain of x.

For example, from
$\forall x$ *eats(Ziggy, x)* we can infer
eats(Ziggy, IceCream)

The variable symbol can be replaced by any ground term, i.e., any constant symbol or function symbol applied to ground terms only.

We can also infer
eats(Ziggy, Peanuts)
eats(Ziggy, Bananas)
…

Existential Elimination/(Exist. Instantiation)

From ∃x P(x) we can infer P(C), where c is a constant that does NOT appear anywhere else in the knowledge base.

C is called a Skolem constant. (Rule is also called Skolemization rule)

For example, from

∃x eats(Ziggy, x) infer eats(Ziggy, C)

Note that the variable is replaced by a brand new constant that does not occur in this or any other sentence in the KB.

In other words, we don't want to accidentally draw wrong inferences about the KB by introducing the constant. All we know is there must be some constant that makes this true, so we can introduce a brand new one to stand in for that (unknown) constant.

Existential Elimination

Another example:

$\exists x \; Crown(x) \wedge OnHead(x, John)$ yields

$Crown(C_1) \wedge OnHead(C_1, John)$

provided C_1 is a new constant symbol (Skolem constant)

Universal & Existential Generalization

Universal generalization

From $P(A) \wedge P(B) \wedge P(C)$ … we can infer $\forall x \, P(x)$,

where A, B, C, ... is the range of all possible constant for the predicate P

Existential generalization

From $P(A)$ we can infer $\exists x \, P(x)$

Reduction to propositional inference

Suppose the KB contains just the following:

$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
$King(John)$
$Greedy(John)$
$Brother(Richard,John)$

Instantiating the universal sentence in all possible ways, we have:

$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
$King(John)$
$Greedy(John)$
$Brother(Richard,John)$

The new KB is propositionalized: proposition symbols are:
King(John), Greedy(John), Evil(John), King(Richard), etc.

Reduction (cont.)

Every FOL KB can be propositionalized so as to preserve entailment

(A ground sentence is entailed by new KB iff entailed by original KB)

Idea: propositionalize KB and

- query,
- apply resolution,
- return result

Problem: with function symbols there are infinitely many ground terms,

- e.g., *Father*(*Father*(*Father*(*John*)))

Reduction (cont.)

Theorem (Herbrand, 1930): If a sentence α is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB

Idea: For $n = 0$ to ∞ do

create a propositional KB by instantiating with depth-n terms
see if α is entailed by this KB

Problem: works if α is entailed, loops if α is not entailed

Theorem: Turing (1936), Church (1936)

Entailment for FOL is semi-decidable (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every non-entailed sentence)

Problems with propositionalization

Propositionalization seems to generate lots of irrelevant sentences.

For example, from:

$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
$King(John)$
$\forall y\ Greedy(y)$
$Brother(Richard, John)$

It seems obvious that Evil(John), but propositionalization produces lots of facts such as Greedy(Richard) that are irrelevant

With p k-ary predicates and n constants, there are $p*n^k$ instantiations!

Unification

We can get the inference immediately if we can find a substitution θ such that

King(x) and *Greedy(x)* match

King(John) and *Greedy(y)*

$\theta = \{x/John, y/John\}$ works

Unify procedure: Unify(P,Q) takes two atomic (i.e. single predicates) sentences P and Q and returns a substitution that makes P and Q identical.

Unifier: a substitution that makes two clauses resolvable.

Unification

Knows(John,x), Knows(John, Jane)

– θ = {x/Jane)

Knows(John,x), Knows(y, Bill)

– θ = {x/Bill, y/John)

Unification

Which unification solves

Knows(John,x), Knows(y, Mother(y))

Choose from:

– A: {y/Bill, x/Mother(John)}
– B: {y/John, x/Bill}
– C: {y/John, x/Mother(John)}
– D: {y/Mother(John), x/John}
– E: {y/John, x/Mother(y)}
– F: {fail}!

What about Knows(John,x), Knows(x, Bill)?

Unification: MGU

To unify *Knows(John,x)* and *Knows(y,z)*,

θ = {y/John, x/z } or θ = {y/John, x/John, z/John}

The first unifier is more general than the second.

There is a single most general unifier (MGU) that is unique up to renaming of variables.

MGU = { y/John, x/z }

Unification – more examples

Find the most general unifier for the sentences below:

- parents(x, father(x), mother(Bill))
  parents(Bill, father(Bill), y)
  - {x/Bill, y/mother(Bill)}

- parents(x, father(x), mother(Bill))
  parents(Bill, father(y), z)
  - {x/Bill, y/Bill, z/mother(Bill)}

- parents(x, father(x), mother(Jane))
  parents(Bill, father(y), mother(y))
  - Failure

Unification

A variable can never be replaced by a term containing that variable.

For example, $\theta = \{x/f(x)\}$ is illegal!

Generalized Modus Ponens

Generalized Modus Ponens:

From

$P(c)$, $Q(c)$ and $\forall x\ (P(x) \wedge Q(x) \Rightarrow R(x))$

we can derive

$R(c)$

Example: From

King(John), Greedy(John) and $\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)

we can derive

Evil(John)

Generalized Modus Ponens

Generalised Modus Ponens (General case): Given

- atomic sentences P1, P2, ..., PN
- implication sentence $(Q1 \wedge Q2 \wedge \ldots \wedge QN) \Rightarrow R$
  where Q1, ..., QN and R are atomic sentences
- substitution subst(θ, Pi) = subst(θ, Qi) for i=1,...,N

we can derive new sentence:

subst(θ, R)

Substitutions

- subst(θ, α) denotes the result of applying a set of substitutions defined by θ to the sentence α
- substitution list θ = {v1/t1, v2/t2, ..., vn/tn} means to replace all occurrences of variable symbol vi by term ti
- substitutions are made in left-to-right order in the list

  E.g. subst({x/IceCream, y/Ziggy}, eats(y,x)) = eats(Ziggy, IceCream)

Genrealized Modus Ponens

Recall our example

$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
$King(John)$
$\forall y\ Greedy(y)$
$Brother(Richard, John)$

And we would like to infer $Evil(John)$ without propositionalization

We can use Generalised modus ponens with the unification:

- P1 is *King*(*John*) Q1 is *King*(*x*)
- P2 is *Greedy*(*John*) Q2 is *Greedy*(*x*)
- θ is {x/John, y/John} R is *Evil*(*x*)
- Subst(θ,R) is *Evil*(*John*)

Implicit assumption that all variables are universally quantified

Completeness and Soundness of GMP

GMP is sound

- Only derives sentences that are logically entailed

(We won't be looking at the proof)

GMP is complete for a KB consisting of definite clauses

- Complete: derives all sentences that entailed
- OR…answers every query whose answers are entailed by such a KB

Definite clause: disjunction of literals of which exactly 1 is positive, e.g.,

$$\neg \text{King}(x) \vee \neg \text{Greedy}(x) \vee \text{Evil}(x)$$

$$\text{King}(x) \text{ AND } \text{Greedy}(x) \Rightarrow \text{Evil}(x)$$

Horn clause

Horn clause – a clause (i.e., a disjunction of literals) that contains at most one positive literal

Horn clauses are usually written as

$$\neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_n \vee Q$$

or the equivalent

$$P_1 \wedge P_2 \wedge \ldots \wedge P_n \Rightarrow Q$$

Or in the case of no positive literal the Horn clause is called goal

$$P_1 \wedge P_2 \wedge \ldots \wedge P_n \Rightarrow \text{false}$$

Definite clause is a horn clause with exactly 1 positive literal

Horn clauses

Not everything can be expressed as a Horn clause. These are not Horn clauses:

- $P(x) \vee Q(x)$
- $(P \wedge Q) \Rightarrow (R \vee S)$

Horn clauses represent a *subset* of the set of sentences representable in FOL – not every FOL sentence is a horn clause

Prolog is based on Horn clauses

Why do we learn about Horn clauses?

Inference using GMP is complete for KBs containing only Horn/definite clauses

Inference approaches in FOL

Forward-chaining
- Uses GMP to add new atomic sentences
- Useful for systems that make inferences as information streams in
- Requires KB to be in form of first-order definite clauses

Backward-chaining
- Works backwards from a query to try to construct a proof
- Can suffer from repeated states and incompleteness
- Useful for query-driven inference

Resolution-based inference (FOL)
- Refutation-complete for general KB
  - Can be used to confirm or refute a sentence p (but not to generate all entailed sentences)
- Requires FOL KB to be reduced to CNF
- Uses generalized version of propositional inference rule

Note that all of these methods are generalizations of their propositional equivalents

Prolog

What is Prolog?

- Programs consist of logical formulas
- Running a program means proving a theorem

Syntax of Prolog

- Predicates, objects, and functions: append(a,b)
- Variables: X, Y, List
- Facts: college(dit).

Prolog

Syntax of Prolog (cont.)

- Rules:

  animal(X) :- cat(X).

  student(X) :- person(X), enrolled(X,Y), university(Y).

- implication ":-" with single predicate on left and only non-negated predicates on the right. All variables implicitly "forall" quantified.

Queries

- student(X)
- All variables implicitly "exists" quantified.

Prolog and Horn clauses

Prolog

```
c:- a, b.
a.
b.
```

Horn clause

$[c \vee \neg a \vee \neg b] \wedge a \wedge b$

$[c \vee \neg a \vee \neg b] \; [a] \; [b]$

FOL: Deducing hidden properties

Back to our Wumpus world example:

$\forall$x,y,a,b *Adjacent*([x,y],[a,b]) $\Leftrightarrow$
[a,b] $\in$ {[x+1,y], [x-1,y],[x,y+1],[x,y-1]}

Properties of squares:
$\forall$s,t *At*(Agent,s,t) $\wedge$ Breeze(t) $\Rightarrow$ Breezy(s)

Squares are breezy near a pit:

- Diagnostic rule - infer cause from effect
  $\forall$s Breezy(s) $\Rightarrow$ Adjacent(r,s) $\wedge$ Pit(r)
- Causal rule - infer effect from cause
  $\forall$r Pit(r) $\Rightarrow$ ($\forall$s Adjacent(r,s) $\Rightarrow$ Breezy(s))

Knowledge engineering in FOL

1. Identify the task
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base

Example: Knowledge Base in FOL

The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

Knowledge Base in FOL

The law says that it is a crime for an American to sell weapons to hostile nations.

The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

... it is a crime for an American to sell weapons to hostile nations:

$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Nono … has some missiles, i.e., $\exists x\ Owns(Nono,x) \wedge Missile(x)$:

$Owns(Nono,M_1)$ and $Missile(M_1)$

… all of its missiles were sold to it by Colonel West

$Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$

Knowledge Base in FOL

Missiles are weapons:

$Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as "hostile“:

$Enemy(x, America) \Rightarrow Hostile(x)$

West, who is American …

$American(West)$

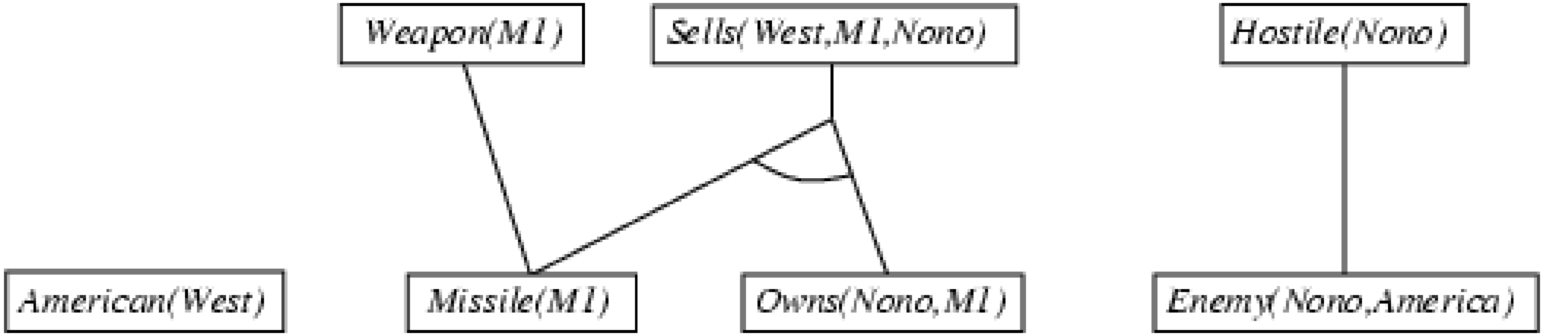
The country Nono, an enemy of America …

$Enemy(Nono, America)$

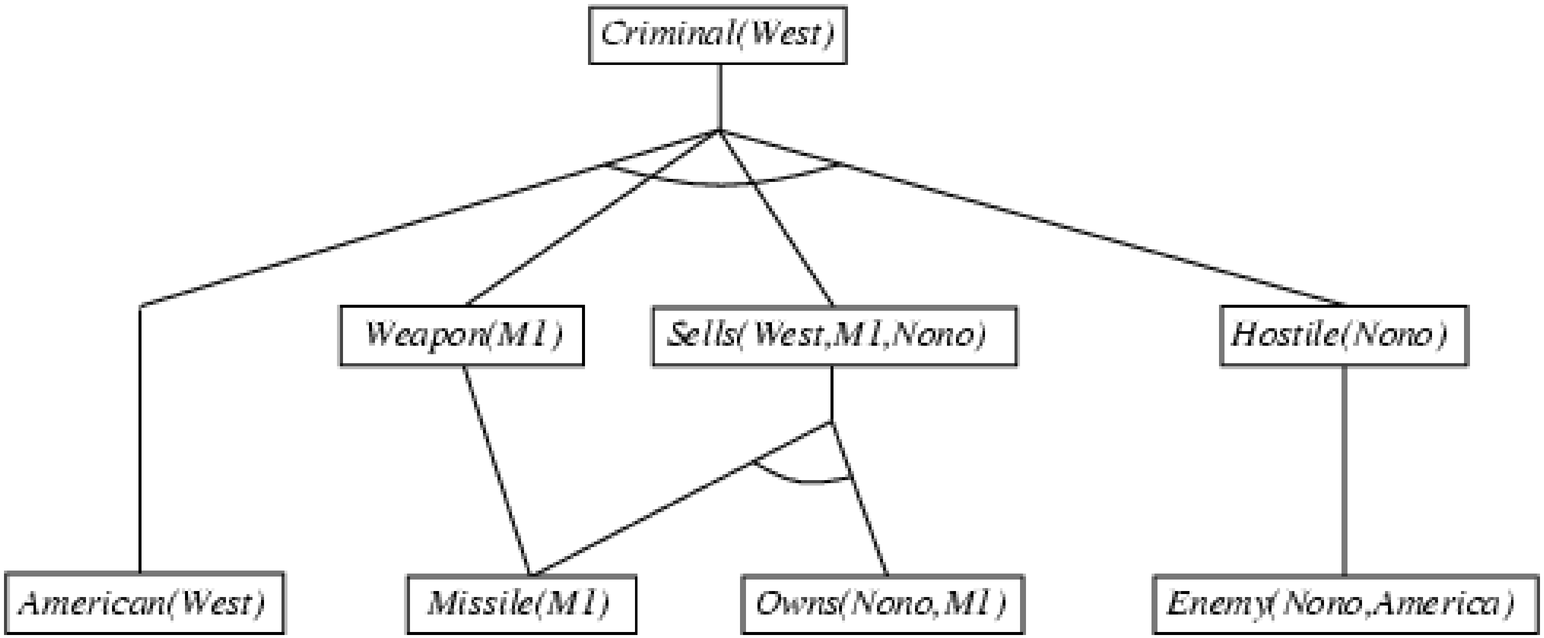
Forward chaining proof

$\boxed{American(West)}$ $\boxed{Missile(M1)}$ $\boxed{Owns(Nono, M1)}$ $\boxed{Enemy(Nono, America)}$

Forward chaining proof



Forward chaining proof



Properties of forward chaining

Sound and complete for first-order definite clauses

Datalog = first-order definite clauses + no functions

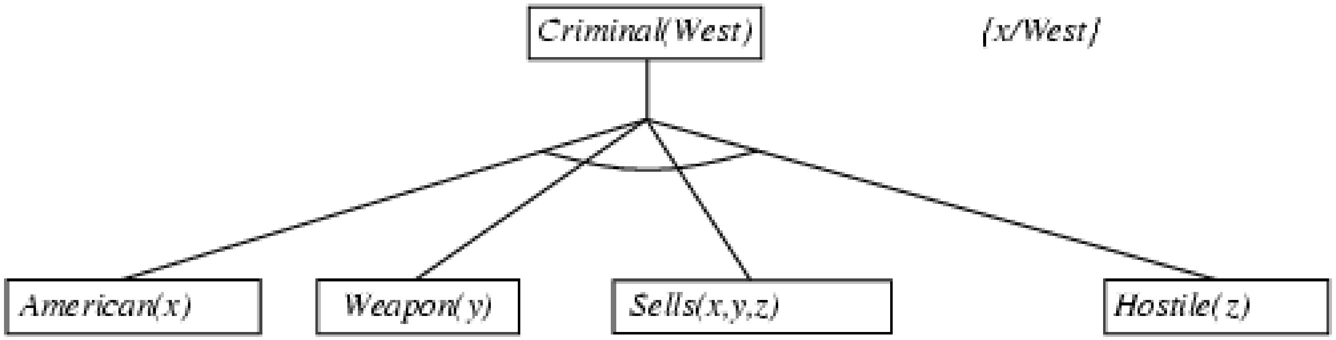
FC terminates for Datalog in finite number of iterations

May not terminate in general if a is not entailed

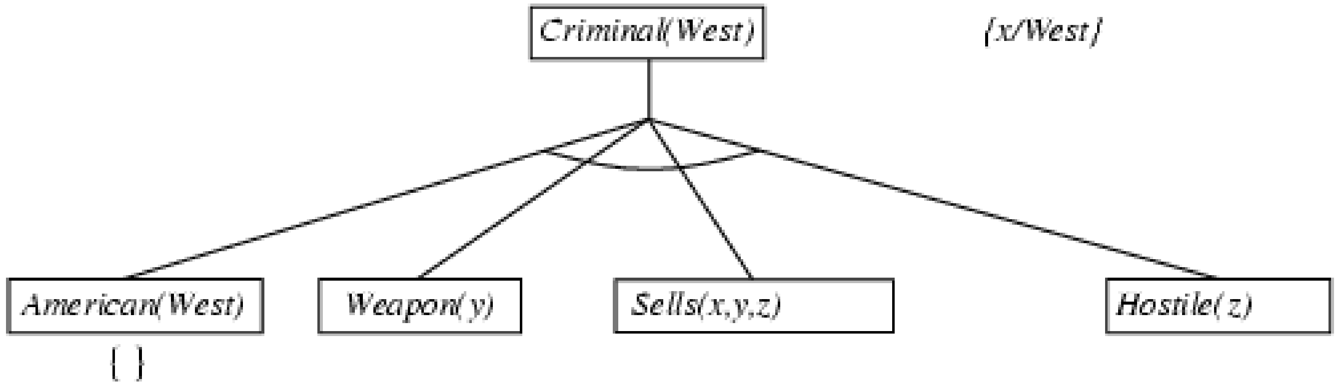
Backward chaining example

$\boxed{Criminal(West)}$

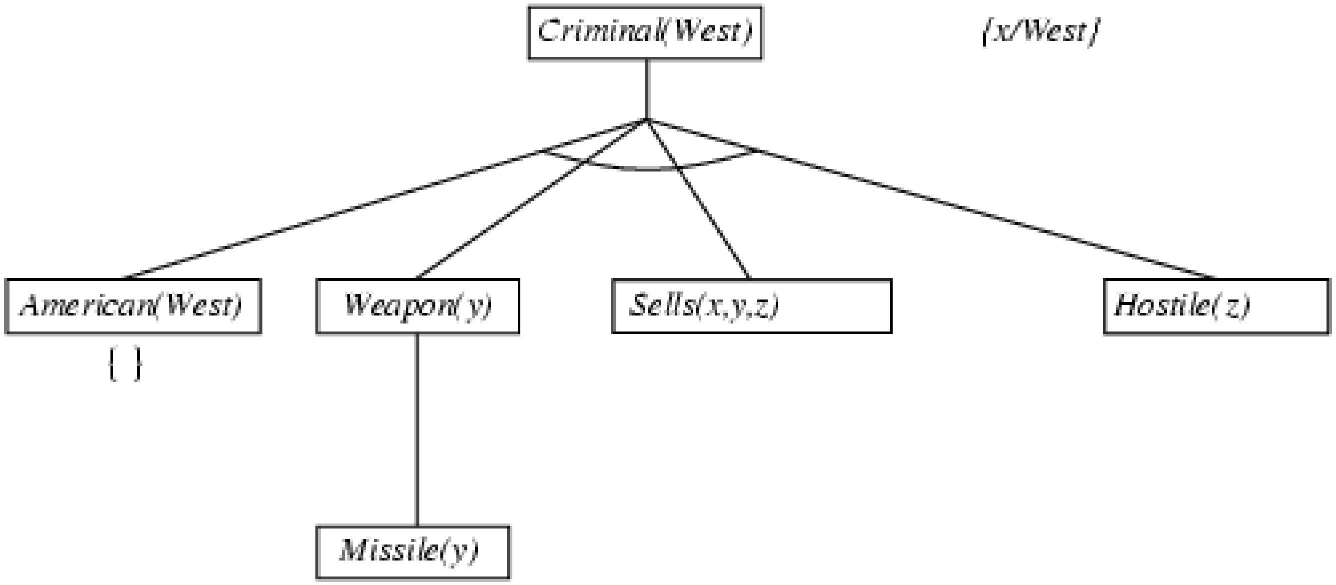
Backward chaining example



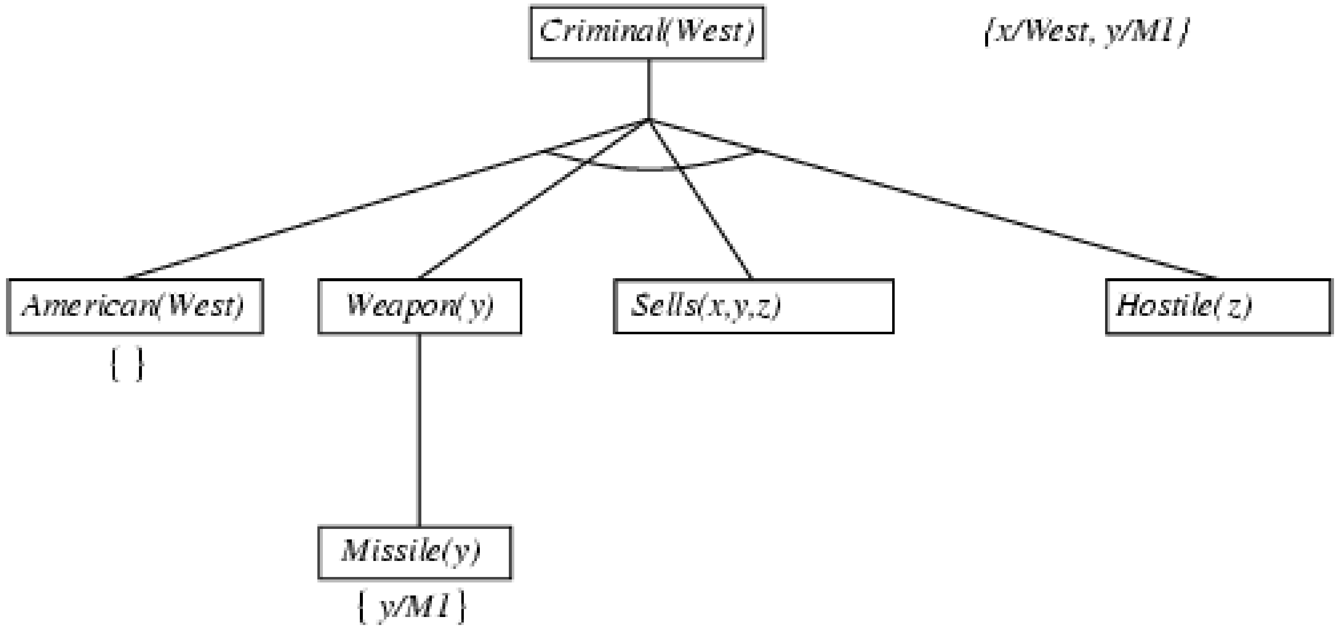
Backward chaining example



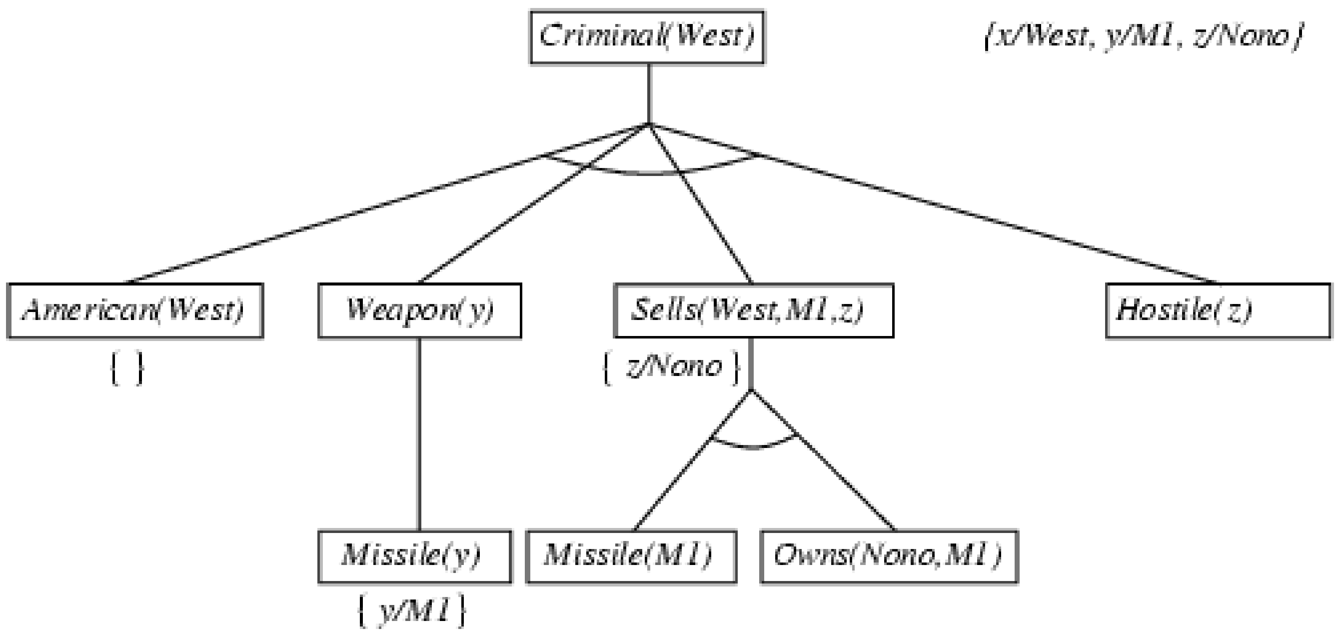
Backward chaining example



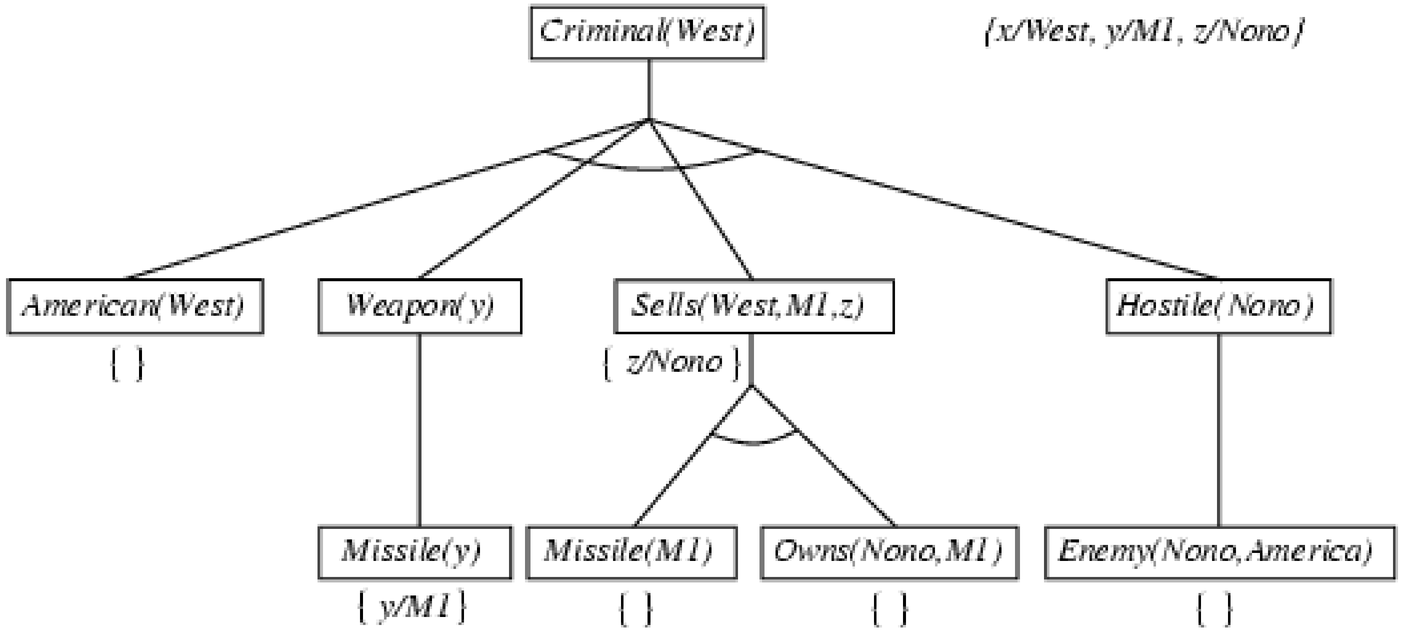
Backward chaining example



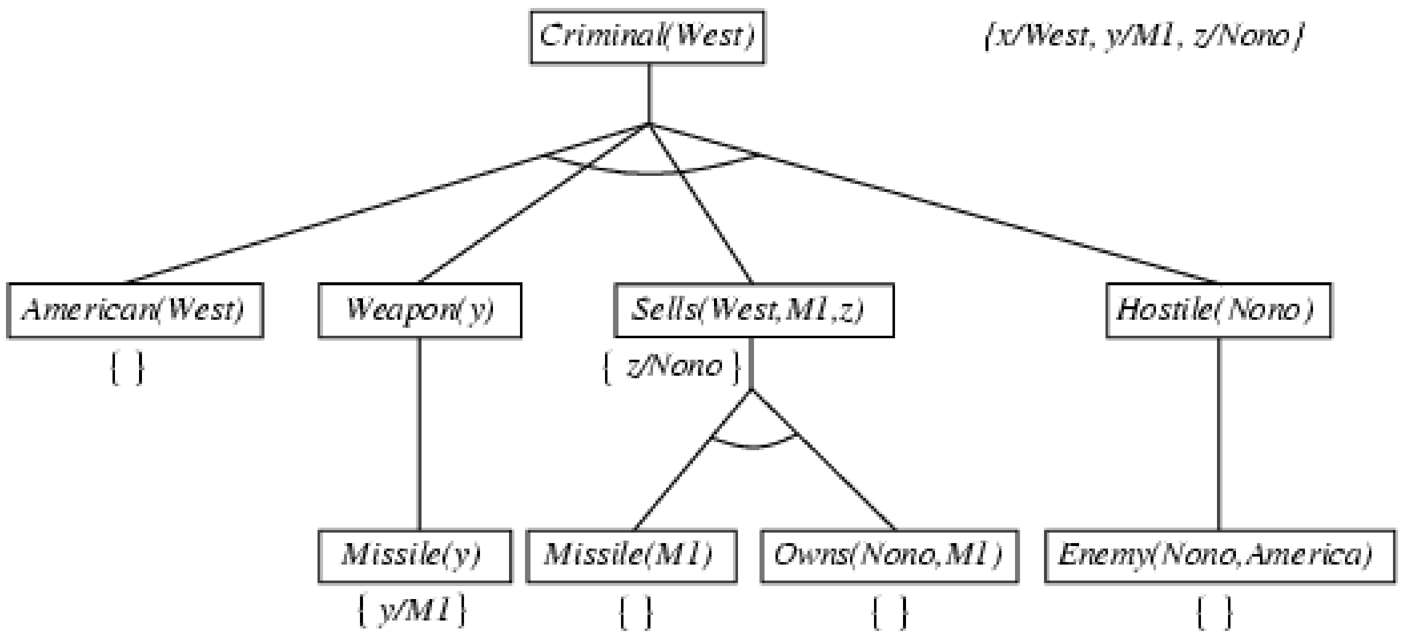
Backward chaining example



Backward chaining example



Backward chaining example



Resolution: brief summary

Full first-order version:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\theta(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)}$$

where $\mathtt{Unify}(\ell_i, \neg m_j) = \theta$.

The two clauses are assumed to be standardized apart so that they share no variables.

For example,

$$\frac{\neg Rich(x) \vee Unhappy(x), \quad Rich(Ken)}{Unhappy(Ken)}$$

with $\theta = \{x/Ken\}$

Apply resolution steps to CNF(KB $\wedge$ $\neg\alpha$); it is complete for FOL

Conversion to CNF

Everyone who loves all animals is loved by someone:

- $\forall x\ [\forall y\ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y\ Loves(y,x)]$

1. Eliminate bi-conditionals and implications
   - $\forall x\ [\neg\forall y\ \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

2. Move $\neg$ inwards: $\neg\forall x\ p \equiv \exists x\ \neg p$, $\neg\exists x\ p \equiv \forall x\ \neg p$
   - $\forall x\ [\exists y\ \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y\ Loves(y,x)]$
   - $\forall x\ [\exists y\ \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$
   - $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

Conversion to CNF (cont.)

3. Standardize variables: each quantifier should use a different one

   $\forall x\ [\exists y\ Animal(y) \land \neg Loves(x,y)] \lor [\exists z\ Loves(z,x)]$

4. Skolemize: a more general form of existential instantiation.

   Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

   $\forall x\ [Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$

5. Drop universal quantifiers

   $[Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$

6. Distribute $\lor$ over $\land$

   $[Animal(F(x)) \lor Loves(G(x),x)] \land [\neg Loves(x,F(x)) \lor Loves(G(x),x)]$

Recall: Example Knowledge Base in FOL

... it is a crime for an American to sell weapons to hostile nations:

American(x) $\wedge$ Weapon(y) $\wedge$ Sells(x,y,z) $\wedge$ Hostile(z) $\Rightarrow$ Criminal(x)

Nono … has some missiles, i.e., $\exists$x Owns(Nono,x) $\wedge$ Missile(x):

Owns(Nono,M_1) and Missile(M_1)

… all of its missiles were sold to it by Colonel West

Missile(x) $\wedge$ Owns(Nono,x) $\Rightarrow$ Sells(West,x,Nono)

Missiles are weapons:

Missile(x) $\Rightarrow$ Weapon(x)

An enemy of America counts as "hostile“:

Enemy(x,America) $\Rightarrow$ Hostile(x)

West, who is American …

American(West)

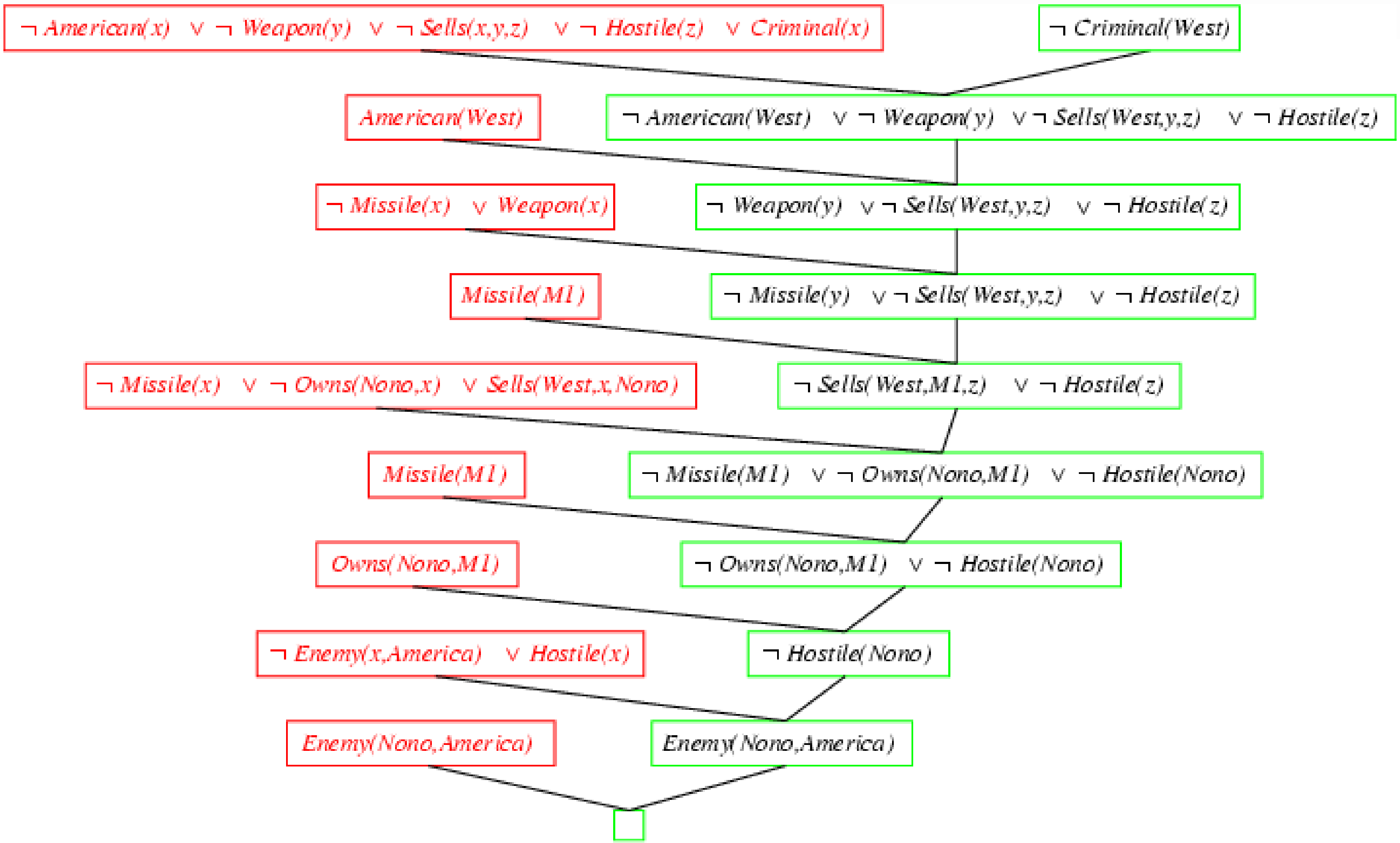
The country Nono, an enemy of America …

Enemy(Nono,America)

Can be converted to CNF

Query: Criminal(West)?

Resolution proof



Another example: *Did Curiosity kill the cat*

Jack owns a dog. Every dog owner is an animal lover. No animal lover kills an animal. Either Jack or Curiosity killed the cat, who is named Tuna. Did Curiosity kill the cat?

These can be represented as follows:

A. $(\exists x)\ Dog(x) \wedge Owns(Jack,x)$

B. $(\forall x)\ ((\exists y)\ Dog(y) \wedge Owns(x, y)) \rightarrow AnimalLover(x)$

C. $(\forall x)\ AnimalLover(x) \rightarrow ((\forall y)\ Animal(y) \rightarrow \neg Kills(x,y))$

D. $Kills(Jack,Tuna) \vee Kills(Curiosity,Tuna)$

E. $Cat(Tuna)$

F. $(\forall x)\ Cat(x) \rightarrow Animal(x)$

G. $Kills(Curiosity, Tuna)$ ⟵ GOAL

Example

Convert to clause form

A1. (Dog(D)) D is a skolem constant

A2. (Owns(Jack,D))

B. (¬Dog(y), ¬Owns(x, y), AnimalLover(x))

C. (¬AnimalLover(a), ¬Animal(b), ¬Kills(a,b))

D. (Kills(Jack,Tuna), Kills(Curiosity,Tuna))

E. Cat(Tuna)

F. (¬Cat(z), Animal(z))

Add the negation of query

¬G: (¬Kills(Curiosity, Tuna))

Example

The resolution refutation proof

R1: ¬G, D, {}	(Kills(Jack, Tuna))
R2: R1, C, {a/Jack, b/Tuna}	(¬ AnimalLover(Jack), ¬ Animal(Tuna))
R3: R2, B, {x/Jack}	(¬ Dog(y), ¬ Owns(Jack, y), ¬ Animal(Tuna))
R4: R3, A1, {y/D}	(¬ Owns(Jack, D), ¬ Animal(Tuna))
R5: R4, A2, {}	(¬ Animal(Tuna))
R6: R5, F, {z/Tuna}	(¬ Cat(Tuna))
R7: R6, E, {}	FALSE

Back to Predicate Calculus…

Proof by contradiction

To prove that KB $\models$ a we try to prove that KB $\wedge\neg$a is unsatisfiable (i.e. implies a contradiction)

We need to convert the knowledge base in Conjunctive Normal Form

Example: Prove that KB = $\{P \Rightarrow Q\}$ does not entail

$\alpha = P \vee Q$

Back to Predicate Calculus…

Proof by contradiction (cont.)

Example: Prove that KB = $\{P \Rightarrow Q\}$ does not entail $\alpha = P \vee Q$

Solution:

1. First we define

$\neg\alpha = \neg(P \wedge Q) = \neg P \vee \neg Q$ (by De Morgan's theorem)

2. Then we can define

$KB \wedge \neg\alpha$: $(P \Rightarrow Q) \wedge (\neg P \vee \neg Q)$

Back to Predicate Calculus…

Proof by contradiction (cont.)

3. Convert to CNF

   (¬P ∨ Q) ∧ (¬P ∨ ¬Q)

4. Apply resolution looking for terms in the format

   (A ∨ B) ∧ (C ∨ ¬B) to produce A ∧ C

   From (¬P ∨ Q) ∧ (¬P ∨ ¬Q) after applying resolution for Q and ¬Q, we can infer ¬P

5. We have not reached contradiction therefore conclude KB does not entail α

Higher-order logic

In FOL, variables can only range over objects

Higher order logic allow us to quantify over relations

- More expressive, but undecidable
- Example:

  "two functions are equal iff they produce the same value for all arguments"

  $\forall f\ \forall g\ (f = g) \Leftrightarrow (\forall x\ f(x) = g(x))$
- Example:

  $\forall r\ \text{transitive}(r) \Rightarrow (\forall x\ \forall y\ \forall z\ r(x,y) \wedge r(y,z) \Rightarrow r(x,z))$

Questions?